Numerical solution of ordinary differential equations (ODE’s)

We spend most of the discussion on first order scalar DE’s, which have the form \( x' = f(t,x) \). A DE is a prescription for how the solution \( x(t) \) is changing. The independent variable is \( t \) and the dependent variable is \( x \). Along with the DE \( x' = f(t,x) \), which specifies how \( x \) is changing with \( t \), we also specify a starting value for \( x \), called an initial condition, given in the form \( x(t_0) = x_0 \). The DE \( x' = f(t,x) \) together with a specified initial condition constitute what is called an initial value problem. An initial value problem, under fairly mild conditions on the function \( f(t,x) \), has a unique solution (which may, however, exist only for \( t \) in some interval containing \( t_0 \) due to the solution becoming infinite, but we won’t worry about that either here.)

In solving DE’s numerically, we generate a sequence of values \( x_0, x_1, \ldots, x_n \) which represent approximations to the exact solution \( x(t_0), x(t_1), \ldots, x(t_n) \) at times \( t_0, t_1, \ldots, t_n \) (of course \( x_0 \) will exactly equal to \( x(t_0) \)). This approach is called discretization. Usually the \( t \)-values \( t_0, t_1, \ldots, t_n \) are equally spaced by a small quantity denoted by \( h \) called the stepsize. This will be the case in our discussion below, unless otherwise specified. Remember that, in principle, knowing the value of \( x(t_k) \) determines exactly the solution for any \( t > t_k \) (and any \( t < t_k \) as well) so that methods in which \( x_{k+1} \) depends only on \( x_k \), are somewhat natural. These are called one-step methods and we treat such methods first.

Other methods we study calculate \( x_{k+1} \) based on \( x_k, x_{k-1}, \ldots, x_{k-m} \). These are called multi-step methods.

In either case, each newly calculated of \( x_k \) is referred to as one step of the method, beginning with the first step, the calculation of \( x_1 \).

Euler’s method: This is the simplest method for approximating the solution of a DE. We note that 

\[
f(t,x) = x'(t) \approx \frac{x(t+h) - x(t)}{h} \quad \text{so} \quad x(t+h) \approx x(t) + f(t,x)h \quad \text{and in particular,} \]

\[
x(t_{i+1}) \approx x(t_i) + f(t_i,x(t_i))h.
\]

This motivates us to approximate \( x(t_i) \) by \( x_i \) as generated by
\[ x_{i+1} = x_i + f(t_i, x_i)h. \] This is Euler’s method. Given \( x_0 \), the subsequent points \( x_1, x_2, \ldots \) are easily calculated.

We consider next how to characterize the error in one of these numerical methods. First there is the local truncation error. The local truncation error is the answer to the question: If \( x_i = x(t_i) \) were exactly true (which it is not), what would be the error in \( x_{i+1} \) as an approximation of \( x(t_{i+1}) \)? In the case of Euler’s method, if \( x_i \) is exact,

\[
x(t_{i+1}) = x(t_i + h) = x(t_i) + x'(t_i)h + \frac{1}{2} x''(\xi)h^2 \quad \text{(from Taylor’s theorem)}
\]

and we can see that \( x_{i+1} = x_i + f(t_i, x_i)h \) differs from \( x(t_{i+1}) \) by \( O(h^2) \). Thus we say that the local truncation error in Euler’s method is \( O(h^2) \), read as ”order \( h^2 \)”. As usual \( O(h^2) \) represents a quantity dependent on \( h \) which is no larger than some constant times \( h^2 \). The analysis of the actual error, called the global error, is more difficult but the following rule of thumb is generally true: The local truncation error represents approximately the error in one step of the numerical method; if \( t_0 = a \) and \( t_n = b \) then the global error in \( x_n \approx x(t_n) \) is about \( n \) times the local truncation error, while \( nh = b-a \), a constant.

So if the local truncation error is \( O(h^k) \) for some \( k \), then the global error will be \( nO(h^k) = O(h^{b-1}) \), i.e. the order is one less. So, for instance, the global error for Euler’s method is \( O(h) \) and Euler’s method is referred to as a ”first order method”.

Higher-order methods: Taylor series method

The Taylor series method is not really a practical method, but rather a benchmark against which higher-order methods are compared, and from which these methods are generated. It uses a Taylor expansion similar to the one above with which we analyzed Euler’s method, but it includes higher derivatives of \( x(t) \). To calculate these higher derivatives we begin from the differential equation:

\[ x'(t) = f(t, x) \]. Then, using the chain rule, \( x''(t) = \frac{d}{dt}[x'(t)] = \frac{d}{dt}[f(t, x)] = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} \frac{dx}{dt} = f_t + f_x f \)
\[
\frac{dx}{dt} \text{ by } f(t, x). \text{ If one wants to continue, it is relatively simple, but messy, to do so:}
\]
\[
x''(t) = \frac{d}{dt}[x''(t)] = \frac{d}{dt}[f_t + f_x f] = \frac{\partial}{\partial t} [f_t + f_x f] + \frac{\partial}{\partial x} [f_t + f_x f] \frac{dx}{dt}
\]
\[
= f_{tt} + f_{tx} f + f_x f_t + [f_{tx} + f_x f + (f_x)^2] f
\]
and so forth.

The Taylor series method replaces the Taylor expansion
\[
x(t_i + h) = x(t_i) + x'(t_i) h + \ldots + \frac{1}{k!} x^{(k)}(t_i) h^k + O(h^{k+1})
\]
with the discretization
\[
x_{i+1} = x_i + x'_i h + \frac{1}{2} x''_i h^2 + \ldots + \frac{1}{k!} x^{(k)}_i h^k
\]
where, in the above notation,
\[
x'_i = x'(t_i, x_i) = f(t_i, x_i), \quad x''_i = x''(t_i, x_i) = f_t(t_i, x_i) + f_x(t_i, x_i) f(t_i, x_i),
\]
and so forth. The above would be referred to as the "\(k\)th order Taylor series method" - it's local truncation error is \(O(h^{k+1})\) and, as explained above, its global error is \(O(h^k)\). Although messy, it is straightforward to calculate \(x_{i+1}\) directly from \(x_i\) using the above formulas.

Important in the subsequent discussion is the second order Taylor series method:
\[
x_{i+1} = x_i + x'_i h + \frac{1}{2} x''_i h^2 = x_i + f(t_i, x_i) h + \frac{1}{2} [f_t(t_i, x_i) + f_x(t_i, x_i) f(t_i, x_i)] h^2
\]
which we abbreviate as
\[
x_{i+1} = x_i + f_i h + \frac{1}{2} [f_t + f_x f] h^2
\]
with the subscript \(i\) indicating that the term to the left is to be evaluated at \((t, x) = (t_i, x_i)\).

In any particular example, it is usually easier to develop a Taylor series method by direct differentiation rather than the use of the formulas above involving the partial derivatives of \(f\). For instance, consider
\[ x' = xe^{-xt} \]

We have
\[ x'' = (x')e^{-xt} + x[e^{-xt}(x' t - x)] = xe^{-2xt} - xe^{-xt}(xte^{-xt} - x) = (x - x^2 t)e^{-2xt} + x^2 e^{-xt} \]

and more derivatives can be calculated if desired.

Modifications of Euler’s method:

We consider two intuitive modifications meant to improve Euler’s method and show that they each result in a second order method. Each of the methods attempts to remedy the fact that Euler’s method is always “behind the curve” in that it uses \( f(t_i, x_i) \) to estimate the derivative of \( x \) over the interval \([t_i, t_i + h]\).

The first method, called the modified Euler method, uses the Euler method to estimate the value of \( x(t_{i+1}) \) and then averages the derivative at \( t_i \) and \( x_i \approx x(t_i) \) with the derivative at \( t_{i+1} \) and the estimated value of \( x(t_{i+1}) \) to obtain an improved estimate for \( x_{i+1} \). This results in:

\[
k = hf(t_i, x_i) \quad \text{(estimates the change in } x \text{ over the next time step)}
\]

\[
x_{i+1} = x_i + \frac{f(t_i, x_i) + f(t_i + h, x_i + k)}{2} h \quad \text{(estimates the derivative as an average)}
\]

The second method is called the midpoint method. Here we take \((t_i, x_i)\) and the estimate of \((t_i + h, x(t_i + h))\) from Euler’s method and evaluate \( x' = f(t, x) \) at the point midway between these two and use this derivative to estimate the change in \( x \). This results in

\[
k = hf(t_i, x_i) \quad \text{(estimates the change in } x \text{ over the next time step)}
\]

\[
x_{i+1} = x_i + f(t_i + h/2, x_i + k/2)h
\]

Both of these methods are order-two methods. We can show that by comparing them to the order two Taylor series method and showing that, when expanded about \((t_i, x_i)\), they match up through the \( h^2 \) term with the order two Taylor series method \( x_{i+1} = x_i + f_i h + \frac{1}{2}[f_i + f_x]h^2 \). First we consider the
modified Euler method:

\[
x_{i+1} = x_i + \frac{f(t_i, x_i) + f(t_i + h, x_i + k)}{2} h = x_i + \frac{f(t_i, x_i)}{2} h + \frac{f(t_i + h, x_i + k)}{2} h
\]

\[
= x_i + \frac{f(t_i, x_i)}{2} h + \frac{1}{2} [f(t_i, x_i) + f_i(t_i, x_i)h + f_i(t_i, x_i)h + O(h^2)] h
\]

Above, we have used the linear approximation of \( f(t_i + h, x_i + k) \) about \( (t_i, x_i) \). We now substitute for \( k \) and obtain

\[
x_{i+1} = x_i + \frac{f(t_i, x_i)}{2} h + \frac{1}{2} [f(t_i, x_i) + f_i(t_i, x_i)h + f_i(t_i, x_i)hf(t_i, x_i) + O(h^2)] h
\]

\[
= x_i + f(t_i, x_i)h + \frac{1}{2} [f(t_i, x_i) + f_i(t_i, x_i)f(t_i, x_i)] h^2 + O(h^3)
\]

which matches the order-two Taylor series method to within the \( O(h^3) \) local truncation error. A similar expansion shows that the midpoint method is also order 2.

Runge-Kutta methods:

Runge-Kutta methods are a class of one-step methods which can be developed for any desired order and involve a sequence of function evaluations involving \( f(t, x) \) near \( (t_i, x_i) \) at each step. They are developed by matching the Taylor series method corresponding to the desired order. The calculations are rather complex. The modified Euler and the midpoint method above are examples of second order Runge-Kutta methods. The most famous and widely used Runge-Kutta method is the so-called classical fourth order Runge-Kutta method, given as follows:

\[
k_1 = hf(t_i, x_i)
\]

\[
k_2 = hf(t_i + h/2, x_i + k_1/2)
\]

\[
k_3 = hf(t_i + h/2, x_i + k_2/2)
\]

\[
k_4 = hf(t_i + h, x_i + k_3)
\]

\[
x_{i+1} = x_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\]
This method has local truncation error $O(h^5)$ and global error $O(h^4)$.

There are many Runge-Kutta formulas. The formulas are not unique even for a given order (e.g. we discovered ourselves two different order-two Runge-Kutta formulas). But the one given is the one best known.

Multistep methods and implicit methods: